

WHAT IS CLAIMED IS:

- 1 1. A method for operating a computer system comprising:
2 receiving in the system a description of a finite state machine including a temporal logic
3 condition; and
4 generating code for emulating the described finite state machine.
- 1 2. The method of claim 1, wherein:
2 the received description comprises at least two state definitions and at least one definition
3 of a transition between states; and wherein
4 the received description comprises a conditional expression associated with a first state of
5 the finite state machine, the conditional expression comprising a first temporal logic
6 condition defined by a first temporal logic operator operating on an event, the
7 conditional expression defining a logical condition for taking a first action specified
8 in the description ; and wherein
9 generating code for emulating the described finite state machine comprises generating
10 code for evaluating the conditional expression during emulation.
- 1 3. The method of claim 2, wherein generating code for evaluating the conditional
2 expression comprises:
3 generating code for declaring a counter variable that is not otherwise specified in the
4 description of the finite state machine;
5 generating code for initializing the counter variable upon entry into said first state;
6 generating code for incrementing the counter variable when said first event occurs;
7 generating code for performing a first test associated with said first temporal logic
8 operator on the counter variable when said first state is active; and
9 generating code for taking a first specified action based on the result of said first test.
- 1 4. The method of claim 3, wherein the conditional expression is part of a conditional action
2 expression in the definition of said first state, and wherein said first specified action is
3 defined in the conditional action expression.

1 5. The method of claim 3, wherein the conditional expression is part of the definition of a
2 transition from said first state to a second state and wherein said first specified action is
3 defined by said transition.

1 6. The method of claim 3, wherein the description of the finite state machine further
2 comprises a second conditional expression associated with a second state of the finite
3 state machine, the second conditional expression comprising a second temporal logic
4 condition defined by a second temporal logic operator operating on said event, the second
5 conditional expression defining a logical condition for taking a second action specified in
6 the description and wherein generating code for emulating the finite state machine further
7 comprises:

8 generating code for initializing the counter variable upon entry into said second state;
9 generating code for performing a second test associated with said second temporal logic
10 operator on the counter variable when said second state is active; and
11 generating code for taking a second specified action based on the result of said second
12 test.

1 7. The method of claim 1, wherein the description of a finite state machine is a graphical
2 description.

1 8. The method of claim 2, wherein said first temporal logic operator operates on an event E
2 and a threshold T and is true when the event E has occurred at least T times during the
3 current activation of said first state.

1 9. The method of claim 2, wherein said first temporal logic operator operates on an event E
2 and a threshold T and is true when the event E has occurred at less than T times during
3 the current activation of said first state.

1 10. The method of claim 2, wherein said first temporal logic operator operates on an event E
2 and a threshold T and is true when the event E has occurred exactly T times during the
3 current activation of said first state.

1 11. The method of claim 2, wherein said first temporal logic operator operates on an event E
2 and a threshold T and is true when the event E has occurred a positive integral multiple of
3 T times during the current activation of said first state.

1 12. The method of claim 7, wherein the graphical representation is a Stateflow® diagram.

1 13. The method of claim 7, wherein the conditional expression is part of a conditional action
2 expression which is graphically represented as a textual expression within a node
3 representing a state of the finite state machine.

1 14. The method of claim 7, wherein the conditional expression is part of the definition of a
2 transition from said first state to a second state and the conditional expression is
3 graphically represented as a textual expression that is proximate to a line connecting
4 nodes representing the first and second states.

1 15. The method of claim 1, wherein the generated code is source code in human readable
2 form.

1 16. A method for operating a computer system comprising:
2 receiving in the system a description of a finite state machine including a temporal logic
3 condition; and
4 emulating the described finite state machine.

1 17. The method of claim 16, wherein
2 the received description comprises at least two state definitions and at least one definition
3 of a transition between states; and wherein
4 the received description comprises a conditional expression associated with a first state of
5 the finite state machine model, the conditional expression comprising a first
6 temporal logic condition defined by a first temporal logic operator operating on an
7 event, the conditional expression defining a logical condition for taking a first action
8 specified in the model; and wherein

9 emulating the described finite state machine comprises evaluating the conditional
10 expression during emulation.

1 18. The method of claim 17, wherein the emulating step further comprises:
2 allocating a counter variable that is not otherwise specified in the description of the finite
3 state machine model;
4 initializing the counter variable upon entry into said first state;
5 incrementing the counter variable when said first event occurs;
6 performing a first test associated with said first temporal logic operator on the counter
7 variable when said first state is active; and
8 taking a first specified action based on the result of said first test.

1 19. A computer programming system, comprising:
2 means for receiving in the system a description of a finite state machine including a
3 temporal logic condition; and
4 means for generating code for emulating the described finite state machine.

1 20. A computer programming system comprising:
2 means for receiving in the system a description of a finite state machine including a
3 temporal logic condition; and
4 means for emulating the described finite state machine.

1 21. A computer programming system, comprising:
2 a graphical user interface for receiving in the system a description of a finite state
3 machine including a temporal logic condition; and
4 a code generator for generating code for emulating the finite state machine.

1 22. A computer programming system comprising:
2 a graphical user interface for receiving in the system a description of a finite state
3 machine including a temporal logic condition; and
4 an interpreter for interpreting the received description to emulate the finite state machine.

1 23. A computer software product residing on a computer readable medium, the software
2 product comprising instructions for causing a computer system to:
3 receive in the system a description of a finite state machine including a temporal logic
4 condition; and
5 generate code for emulating the described finite state machine.

1 24. A computer software product residing on a computer readable medium, the software
2 product comprising instructions for causing a computer system to:
3 receive in the system a description of a finite state machine including a temporal logic
4 condition; and
5 emulate the described finite state machine.

1 25. A computer programming system comprising:
2 a central processing unit;
3 a mass storage subsystem;
4 a program editor capable of receiving from a user a description of a finite state machine
5 including a temporal logic condition and storing the description on the mass storage
6 subsystem;
7 a code generator capable of receiving the stored description and generating code for
8 emulating the described finite state machine.

1 26. A computer programming system comprising:
2 a central processing unit;
3 a mass storage subsystem;
4 a program editor capable of receiving from a user a description of a finite state machine
5 including a temporal logic condition and storing the description on the mass storage
6 subsystem;
7 an emulator capable of receiving the stored description and emulating the described finite
8 state machine.